

## d-tree Version 3.1 Release E2 Update Information

The following notes refer to new features that have been added to d-tree since release D2. These changes may not yet be noted in the documentation.

### 1) Field Range Checking

The ability to place a RANGE check on a field has been added to the EDIT ability. This was done by enhancing the TABLE edit type. Along with single values, ranges can be defined in a [a-z] format as elements of the table. Ranges are supported for all field types except strings and dates. Refer to the file named "refcard.doc" on distribution disk for more information on range syntax.

```
EDITS(master)
Must Enter Field          act_code MANDATORY
This record Already Exists act_code DUPKEY act_code_idx
this entry not correct    act_bal  TABLE  1.00 3.00 [100.00-200.00]
this entry not correct    act_char TABLE  A B C [M-O]
this entry not correct    act_uchar TABLE  1 2 3
this entry not correct    act_code TABLE  AA BB CC
```

Ranges Are now Allowed in TABLE Edits  
See the file "refcard.doc" on the distribution  
diskettes for more information.

### 2) Integer Date Field Support

d-tree now properly handles dates that are stored as long integers (i.e.: RTDATE type). The logic to handle the conversion from integer to string and back again is in place. This allows dates to be presented on an IMAGE in a string format such as MM/DD/YY and then be stored as a long integer. All reasonable date formats are supported, including a four digit year. The format of the string is required in order to properly convert the date into a long integer. (i.e.: DD/MM/YYYY as opposed to YY-MM-DD). This format is defined using input attribute field masks within the FIELD ability section. The new mask characters of M,D,and Y have been added to provide this support.

```
IMAGE(master) (LSTFLD_ADVANCE)
```

```
Date: _____
```

Date Input field on screen.  
Defined RTDATE type

```
FIELD(master)
```

/* Symbol Name	Input Attribute	Output Attribute	Input Order	/*
date_field	NONE (YYYY/MM/DD)	NONE	1	/* Date /*
	{MM-DD-YY}			
	{MM-DD-YYYY}			
	{DD/MM/YY}			

Any Combination with year either YY or YYYY.

When the date (in string form) is entered from the screen, the new date input masks are used to convert the entry into a long integer.

### 3) EDITS: MANDATORY\_IF and "OR"ing capability

Three New features have been added to the EDITS ability:

A new edit type of **MANDATORY\_IF** provides a way to indicate that a field is mandatory if another field is not blank (ie: if one field is entered the other is mandatory). To define which field will be checked for an entry, you enclose its symbolic name in brackets {} following the **MANDATORY\_IF** keyword. See illustration below.

The means to "OR" edits has been added to the EDITS ability. Placing the keyword **"OR\_WITH\_NEXT"** as the error message for an edit, specifies that this edit is to be "OR'd" with the next edit.

The new keyword **"LOOKUP\_ONLY"** has been added to the **VALIDATE** edit. Placing this keyword after the map and scan identifiers provides the same lookup capability as a regular **VALIDATE** edit, but relaxes the "equal hit" edit. This allows a key that supports duplicates, to be used for "lookup" purposes.

EDITS(master)	symbol name	edit type	edit information
/* Error Message			
Must Enter Complete Code	cou_cod	MAND_FILL	
Must Enter County Code	cou_cod	MANDATORY_IF {field_symbol}	
This county already exists	cou_cod	DUPKEY C_Num	
OR_WITH_NEXT	cou_cod	VALIDATE ex1idx	cmap cscann prefix
Code must exist in File	cou_cod	VALIDATE ex2idx	cmap cscann prefix
None	cou_cod	VALIDATE ex3idx	map scan LOOKUP_ONLY

#### New Features in EDITS:

- a) a new edit type of **MANDATORY\_IF** has been added. This field will be **MANDATORY** if there is a value in the field supplied in brackets {}.
- b) There is now an "OR" capability in edits. Place the keyword **"OR\_WITH\_NEXT"** as the error message for an edit that is to be or'd with the next edit. "OR" edits may be stacked, with the first edit without the **"OR\_WITH\_NEXT"** keyword as the final or condition. The error message for this edit is displayed if all "OR"'s fail.
- c) The **"LOOKUP\_ONLY"** keyword will relax the "equal hit" edit, while allowing a "lookup" capability. The error message is necessary, although will not be displayed.

### 5) DEFAULTS: List type default ; field default values

Two new features have been added to the **DEFAULTS** ability:

- a) A new defaults type has been added known as a default list type. The best way to describe this is through an example. We want to present the user with a field that has a limited amount of valid entries. In this case we'll use a field where the user is to specify a baud rate. Valid values are limited to 300, 1200, 2400, 9600, and 38400. We could simply place a **TABLE** edit on this field, but this would necessitate the user enter the value. Rather, we would prefer to default the field with a valid value, and allow the user to simply hit the space bar

to change the value to another valid entry. In this case we will default the field to 300 baud, and each time the user hits the space bar the value will change from 300 to 1200; from 1200 to 2400; from 2400 to 9600...from 34400 to 300...etc. See the illustration below. Note: an input attribute of NOCHANGE must be placed on the field in order for the "space bar change logic" to take effect. This is done in the associated FIELD section.

b) Before this release, the value used to default a field could only be a literal keyed in the d-tree script. We have now added new support where the value used to default a field can be the contents of another field. This is done by placing the symbol name of the other field (within brackets with no spaces) as the default value. See illustration below:

```

DEFAULTS(master)
/* Symbol Name      Type of defaults      Defaults value */
office              DFALT_KEY              Reno Office
office_date         INIT                   SYSDATE
office_time         DFALT_KEY              SYSTIME
baud_rate           DFALT_LIST             300 1200 2400 9600 38400
office_name         DFALT_KEY              (office) /* put field symbol in ( ) */

```

Two New DEFAULT features:

- a) Enter a list of valid values. The field will be initialized with the first value. When users enter this field, they can hit the space bar. The value of the field will change to the next entry in the list each time the space bar is hit.
- b) The default value can now be another field value. Simply place the desired field's symbol name within brackets ( ) without any spaces. When the default is executed, the value of current value of this field will be "defaulted" into your destination.

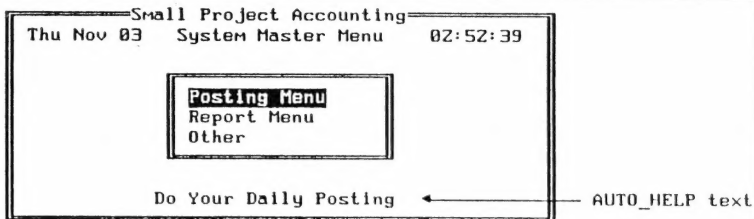
## 4) Menu Enhancements

Three new features have been added to the menu support.

The first two features pertain to cursor movement (option selection) for lotus style menus.

- a) The user now can hit the space bar to move to the next option.
- b) Support to allow the user to enter the first character of a desired option has been added by means of the new input attribute GOTO in the FIELD ability.

The third new feature may be illustrated as follows: While on a certain menu option, it may be helpful to display "help" information pertaining to the option. This "help" should automatically change when the user moves from option to option. This effect is provided by the new AUTO\_HELP input attribute in the FIELD ability working in conjunction with the HELP ability. Place the AUTO\_HELP attribute on the desired field. Add the "help text" via the help ability as shown below. When the cursor enters this field, the help text will automatically be displayed, without the need for the user to hit the help key.



- a) On a Lotus style menu you may now hit the space bar to go to next option  
Example: Hit space bar will move BAR to Report Menu.
- b) Using the new GOTO input attribute allows jumping to options by entering the first letter of the option. Example: Hitting "O" will take us to "Other" option.
- c) Using the new AUTO\_HELP input attribute allows option description text to be displayed while on an option. As you move from option to option this text will change.

## FIELD(menu)

Symbol Name	Input Attribute	Output Attribute	Input Order
menu1a	GOTO AUTO_HELP NOCHANGE	INPUTRI	1
menu1b	GOTO AUTO_HELP NOCHANGE	INPUTRI	2
menu1c	GOTO AUTO_HELP NOCHANGE	INPUTRI	3

- b) Allow user to hit first character of another menu option to GOTO that option.
- c) Display Help text automatically when entering this help.

## HELP(menu)

Help Text	Field Symbol
Do Your Daily Posting	menu1a
Go To Report Menu	menu1b
Describe Other Here	menu1c

Help text defined here.

## 6) Prompt Ability

The scan name was mandatory in the PROMPT ability syntax in the prior release. The DT\_PRMP function can be very useful in creating key targets, without the use of a scan. Therefore we have allowed the keyword of NONE to be used as a scan name in the PROMPT ability syntax.

## PROMPT(master)

## USES\_IMAGE(prompt)

key symbol name	scann name	fields for target	prefix
C_Num	NONE	cou_cod	
C_Nam	master	cou_name	
NONE	master	option	

No scann defined.

## 7) IFILS: DODA can be defined in script

d-tree now supports the ability to define the fields in the data object definition array (DODA) through a d-tree script, as opposed to being hard coded within the program. This is done by use of the new "PGM\_FIELDS" keyword within the IFILS ability section. Simply enter your program fields (DODA elements) following the syntax illustrated below. At this point in time, d-tree does not support part of the DODA being hard coded, while other parts are dynamic (defined in a d-tree script). The ability to define fields in a dynamic manner is a powerful concept. The field types follow r-tree's naming convention and can be found in the file "rtdoda.h".

### IFILS

#### PGM\_FIELDS

/* doda symbol name	field type	field length	*/
name	STRING	11	
address	STRING	30	
balance	RTDFLOAT	8	
quantity	RTINTZ	2	

d-tree now supports the ability to define a DODA from a d-tree script. This is done with a new keyword that has been added to the IFILS ability. By entering PGM\_FIELDS within the IFILS ability section the DODA or in other words, program fields can now be defined.

**IFILS Change:** The "DICTIONARY" capability discussed in the IFILS section in the d-tree manual has been changed. There is no longer a "DICTIONARY" keyword within the IFILS ability. Due to the power and flexibility that can be obtained by accessing definitions from a data dictionary, this capability has been moved to its own ability type. Access to the data dictionary at runtime can now be obtained by means of the new DATA\_DICTIONARY ability. See below.

## 8) New Data Dictionary Ability

A new ability to allow file definitions to be defined from within a d-tree script has been added. By following the syntax shown below, in conjunction with the new function call DT\_DDICT, file definitions can be accessed at runtime.

```
DATA_DICTIONARY(myfiles)
```

```
DICTIONARY diction.dta
FILE_NAME  filename.dta
VERSION    1.0
```

File definitions used in a program can now be defined from a d-tree script. A new d-tree function called DT\_DDICT(ref) has been added to the toolbox. By calling this function from within your program, file and index definitions (this includes IFIL, IIDX, ISEG, and DODA definitions) are swapped into memory at runtime. Example: Consider the definition above in your d-tree script and the following line of code in your c program.

```
if (uerr_cod=DT_DDICT(DT_INAME("myfiles"))))
    printf("Could no get file definition from data dictionary");
```

DT\_DDICT will open the data dictionary with the file name of "diction.dta". It will then try to read in the definitions for file "filename.dat" version 1.0. (data dictionary stores multiple file definitions with version control)

## 9) Parsing Error Help.

When an error occurs while parsing a d-tree script, the ability name, as well as the ability number will now be displayed. If you started at the top of the script and counted all ability definition sections (count all ability types, not just the type that had the error), until you hit the ability number shown, you would be on the section where the error occurred.

```
C:\DT >posting ← Run your Program.
token='tamntx' ← Last Token hit in script.
               ↓ Ability that was being parsed.
```

```
<FIELD> (the #7 ability in the script) has an Error
Error Occurred During Parse Error=1001hex 4106
```

```
C:\DT >           ↑ c-tree error or
                  ↑ d-tree error
```

## 10) GROUPS - New High Level Group In/Out.

The function DT\_GPHIN (group high level in) and DT\_GPHOT (group high level out) have been added to d-tree. DT\_GROUT provides a means to store a script definition into a "group" file. DT\_GPHIN is used to access the definition and swap it back into memory. See DT\_GROUT.C and DT\_GROUP.C.

## 11) DT\_FUNCP - Set Function pointers

As we begin work to bring the code size down in d-tree, we have started to use a number of function pointers. These pointers are initialized for you by d-tree when ability definitions are being parsed in, or they are hard coded. If ability definitions are being swapped in through the use of GROUPS, you are responsible for initializing these pointers. The function DT\_FUNCP has been provided to perform this initialization. This function must be called at the top of your program just after DT\_SETTY to set the function pointers. **THIS ONLY APPLIES IF YOU ARE USING GROUPS.** Note: The current version of this function initializes all pointers. You may want to create your own version of this function, only specifying the functions pointers your program will be utilizing, in order to decrease code size. The function can be found in the source file DT\_FUNCT.C.

## 12) Easy way to rebuild indices from the catalog

We have added an easy way to provoke a file rebuild. The Reformat option on the catalog's data dictionary menu has been enhanced. Now called the "Reformat/Rebuild" option, this selection allows files to be chosen for rebuild. By rebuild we mean: c-tree's RBLFIL will be called for each file chosen. Simply place an "R" next to the file to be rebuilt as shown below:

Sun Nov 06		FairCom		02:59:41
File Reformatting/Rebuild Utility				
Sel Name	Version	Description	System	
account	1	Account Master	MISC	
[R]customer	1	customer	IMPORTED	
distribut	1	Distribution Detail File	MISC	
R project	1	project	IMPORTED	
transact	1	Transaction File	MISC	
R vendor	1	Vendor Master	MISC	

The customer, project, and vendor files will be rebuilt. For each file chosen, c-tree's RBLFIL will be called to rebuild file header information, as well as reconstruct the file indices.

Enter (T)o / (F)rom for Reformatting Data or (R)ebuild to rebuild indices  
Press 'ESC ESC' to EXIT or 'POST' key to begin Reformat/Rebuild

### **13) Running Clock (DOS INSTANT ONLY)**

We have enhanced the time display to now show a running, 12 hour clock. This is only supported in the DOS environment when using INSTANT screens. We hope this is not a limitation in future releases.

### **14) DT\_INIMG - Initialize fields for a certain IMAGE**

The function DT\_INIMG function has been added to allow initialization of all fields that pertain to a specific IMAGE. Example:

```
DT_INIMG(1);
```

will look for all fields on IMAGE number 1 and initialize them. This function may be found in the source module DT\_IMAGE.C.

### **15) DT\_TXFLD - return the value a field in text form.**

The function DT\_TXFLD has been added for the following purpose: Given a text pointer, followed by a FIELD pointer (type DTTFIELD), this function will place the value of the field into the buffer pointed to by the text pointer. This value will be in the same form as if it were sent to the screen. (masks included, numeric to ascii, etc.).

### **16) SCAN Ability - Bar Select and Add Mode.**

Two very nice features have been added to the SCAN ability. The most current scan module can be found in the "patches" directory on disk #5. Because of the usefulness of the following features, we pushed to get them in this release. Although the features are complete, we have not thoroughly tested them in all environments. The previous version of the DT\_SCANN.C module can be found on disk #3 if found necessary. We suggest starting with the latest (disk #5 version of DT\_SCANN.C) which contains the following features:

First the ability to define an alternate way to select a record. As opposed to the user being displayed a list of numbered records and having to enter the number of the desired record, d-tree now supports defining a "SELECT BAR". This means that records can be selected by positioning a reverse image bar on the desired record and pressing 'return'. This approach is supported as follows: a) the image reference provided for the IMAGE\_INP definition should be the same as the IMAGE\_ROL definition. b) The new keyword "SELECT\_BAR" must be added to the SCAN definition section. See illustration.



Sun Nov 06	MISC System Transaction File	04:42:17
reference number:	112200	
type:	C	
date:	10/31/88	
amount:	1000.00	
description:	Another Modem Phone Bill	
customer/vendor code:		

Breakdown Amt	Account	Project
Chart of Accounts	?	
101 Cash In Bank		
102 Travel		
103 Entertainment		
<b>104 Phone Expense</b>		
105 Equipment		
106 Utilities		
Press ESC ESC to EXIT		

A lookup was provoked.  
A POP-UP selection window is displayed with Page Up/Down and Up/Down Keys supported to find selection.

Trn No: 1

The script to provide this selection looks like this:

```

posting.dts
IMAGE(acthead) {POP_UP} {BASE_ROW=12}
+Transaction File

                                     *Note the highlighted keywords

Press ESC ESC to EXIT

+

IMAGE(actroll) {CLR_BLOCK} {BASE_ROW=12} {LSTFLD_ADVANCE} {FRSFLD_BACKUP}

FIELD(actroll)
/* Symbol Name      Input Attribute  Output Attribute  Input Order  I/O Special */
acode               NOCHANGE         NONE             1 /* account code */
adesc               PROTECT          NONE             2 /* desc */

SCAN(actscan) {IMAGE_OUT=acthead} {IMAGE_ROL=actroll} {IMAGE_INP=actroll}
               {SELECT_BAR} {ROLL_LINES=6}

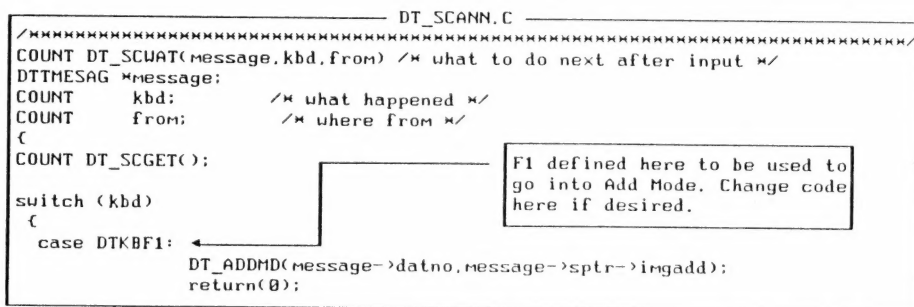
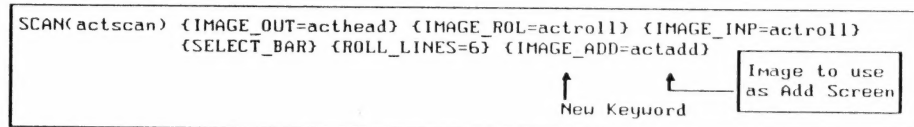
```

The scan ability still supports selecting records by entering a reference number when the image references for IMAGE\_ROL and IMAGE\_INP are different. We have enhanced this selection. You may now enter either the record's reference number, or point to the record for selection. The new point feature (as well as the character used to "point") are defined by the `#define SCAN_POINT` at the top of DT\_SCANN.C. Commenting out this `#define` will relax the "point" character and "point" selection ability.

Sun Nov 06		MISC System Transaction File		04:41:31
Select				
1 01	111111	11	1	
2 02	222222	2	2	
3 03	3	3	3	
4 04		4	4	
5 05	5	5	5	
7 07	7	7	7	Point Character as defined by SCAN_POINT in DT_SCANN.C. Can hit 'RETURN' to select record #0.
8 08	8	8	8	
9 09	9	9	9	
10 10	10	1	10	
11 11	1	1	11	
12 12			12	
13 13			13	
14 14			14	
15 15			15	
16 16			16	
17 17			17	
18 18			18	

Enter Desired Option:[\_]   
 Press ESC ESC to EXIT   
 FairCom (c) 1988

The second feature allows the user to enter into an "ADD MODE" for the file that is being scanned. Example: Say we are entering a customer number when keying invoices. We enter a "?" in the customer id field to do a lookup into the customer file. We discover that this is a new customer who should be added to the customer master file. While in the scan screen, d-tree now supports the ability to enter an "ADD MODE" into the customer file. The new scan keyword "IMAGE\_ADD" has been added for this purpose. Providing the image to be used for the "ADD", will allow the user to hit "F1" from the scan screen. This will provoke the add process, which, once complete, will return to the scan. See illustration below:



## BUGS FOUND

The following list contains areas where bugs were found. If you were experiencing problems in any of these areas, we hope we've solved the problem.

- Default "AUTO DUP" feature.
- Numeric input of fields(delete key problem).
- System hanging when input masks were defined for numeric fields.
- TABLE edit problem.
- IMAGE parsing problems (TABS in IMAGE now ok).
- r-tree front end prompt parsing error.
- Window overlapping problems.
- Ending comment left out of code.
- Unix/Xenix termio problem.
- Help ability bug.
- Page Up/Down problem in subfiles.
- Hard coded hooks problem.
- Subfile map problem.
- Hard coded input fields.

## CATALOG

The Catalog program had a number of bugs that were fixed, primarily where "core dumps" where occuring in the Xenix/Unix world. Below are some of the more obvious problems that have been fixed:

- **RECORD LENGTHS:** When a field length was entered into the data dictionary, the record length was being calculated based on this field length without taking the field type into consideration. I.E: user keys an integer field and says it is 5 long. The length of 5 was being used not only to control the input length that was used when creating a d-tree script , but also as the length to determine record length. The result was invalid record lengths.

Now the catalog acts as follows:

For all field types except char arrays(strings): the length keyed in the data dictionary will be the length used to draw input lines ( \_\_\_\_ ) when creating scripts. The length used to calculate record lengths will be based on the field type, not on the field length. For char arrays(string):

then length will be the length used to calculate record length, and the input lines ( \_\_\_ ) placed in a d-tree script will be one less than the field length. The last byte of a char array is used for the NULL byte.

- User Defined Segment Modes.
- To/From Reformat File Problems.
- Zero Length Parameter File being Created from the catalog.
- Text In/Out Problem.
- Creating default d-tree script: decimal position now added to script.

We would like to thank all the users who have been so helpful in both pointing out problem areas, as well as coming up with good new ideas. We are not going to sit here and take all the credit for the new features that have been implemented in this release. (we will take blame for the bugs in the last release). It's your positive feedback that makes the coding worthwhile. We wish we could have implemented all the useful suggestions, but as you well know, it had to be cut off sometime. Thank you again, and please continue to communicate bugs, suggestions and/or enhancements.

For your information, the areas at the top of the development list are:

- 1) Smaller Code Size.
- 2) Memo Fields (variable length word processing like fields).
- 3) Group Examples.

We list these for information only. We hesitate to give time frames. We encourage users who are in need of a feature to get advice from us as to how to approach the problem, and not to wait on us. We can only promise consistent hard work in our development lab.